Отчёт по лабораторной работе №4

дисциплина: Архитектура компьютера

Клименко Алёна Сергеевна

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Цель данной лабораторной работы - освоить процедуры компиляции и сборки программ, написанных на ассемблере NASM.

2 Задание

1. Создание программы Hello world!
2. Работа с транслятором NASM
3. Работа с расширенным синтаксисом командной строки NASM
4. Работа с компоновщиком LD
5. Запуск исполняемого файла
6. Выполнение заданий для самостоятельной работы.

Основными функциональными элементами любой ЭВМ являются центральный процессор, память и периферийные устройства. Взаимодействие этих устройств осуществляется через общую шину, к которой они подключены. Физически шина представляет собой большое количество проводников, соединяющих устройства друг с другом. В современных компьютерах проводники выполнены в виде электропроводящих дорожек на материнской плате. Основной задачей процессора является обработка информации, а также организация координации всех узлов компьютера. В состав центрального процессора входят следующие устройства: - арифметико-логическое устройство (АЛУ) — выполняет логические и арифметические действия, необходимые для обработки информации, хранящейся в памяти; - устройство управления (УУ) — обеспечивает управление и контроль всех устройств компьютера; - регистры — сверхбыстрая оперативная память небольшого объёма, входящая в состав процессора, для временного хранения промежуточных результатов выполнения инструкций; регистры процессора делятся на два типа: регистры общего назначения и специальные регистры. Для того, чтобы писать программы на

ассемблере, необходимо знать, какие регистры процессора существуют и как их можно использовать. Большинство команд в программах написанных на ассемблере используют регистры в каче- стве операндов. Практически все команды представляют собой преобразование данных хранящихся в регистрах процессора, это например пересылка данных между регистрами или между регистрами и памятью, преобразование (арифметические или логические операции) данных хранящихся в регистрах. Доступ к регистрам осуществляется не по адресам, как к основной памяти, а по именам. Каждый регистр процессора архитектуры x86 имеет свое название, состоящее из 2 или 3 букв латинского алфавита. В качестве примера приведем названия основных регистров общего назначения (именно эти регистры чаще всего используются при написании программ): - RAX, RCX, RDX, RBX, RSI, RDI — 64-битные - EAX, ECX, EDX, EBX, ESI, EDI — 32-битные - AX, CX, DX, BX, SI, DI — 16-битные - AH, AL, CH, CL, DH, DL, BH, BL — 8-битные

Другим важным узлом ЭВМ является оперативное запоминающее устройство (ОЗУ). ОЗУ — это быстродействующее энергозависимое запоминающее устройство, которое напрямую взаимодействует с узлами процессора, предназначенное для хранения программ и данных, с которыми процессор непосредственно работает в текущий момент. ОЗУ состоит из одинаковых пронумерованных ячеек памяти. Номер ячейки памяти — это адрес хранящихся в ней данных. Периферийные устройства в составе ЭВМ: - устройства внешней памяти, которые предназначены для долговременного хранения больших объёмов данных. - устройства ввода-вывода, которые обеспечивают взаимодействие ЦП с внешней средой.

В основе вычислительного процесса ЭВМ лежит принцип программного управления. Это означает, что компьютер решает поставленную задачу как последовательность действий, записанных в виде программы.

Коды команд представляют собой многоразрядные двоичные комбинации из 0 и 1. В коде машинной команды можно выделить две части: операционную и адресную. В операционной части хранится код команды, которую необходимо

выполнить. В адресной части хранятся данные или адреса данных, которые участвуют в выполнении данной операции. При выполнении каждой команды процессор выполняет определённую последовательность стандартных действий, которая называется командным циклом процессора. Он заключается в следующем: 1. формирование адреса в памяти очередной команды; 2. считывание кода команды из памяти и её дешифрация; 3. выполнение команды; 4. переход к следующей команде.

Язык ассемблера (assembly language, сокращённо asm) — машинно-ориентированный язык низкого уровня. NASM — это открытый проект ассемблера, версии которого доступны под различные операционные системы и который позволяет получать объектные файлы для этих систем. В NASM используется Intel-синтаксис и поддерживаются инструкции x86-64.

3 Выполнение лабораторной работы

3.1 Программа Hello world!

В домашней директории создаю каталог, в котором буду хранить файлы для этой лабораторной работы. (рис. 3.1)

Создание рабочей директроии

Рис. 3.1: Создание рабочей директроии

Далее создаю файл hello.asm, в котором буду писать программу на языке ассемблера. (рис. 3.2)

Создание .asm файла

Рис. 3.2: Создание .asm файла

С помощью редактора mousepad пишу программу в созданном файле. (рис. 3.3)

Редактирование файла

Рис. 3.3: Редактирование файла

3.2 Транслятор NASM

Компилирую с помощью NASM свою программу. (рис. 3.4)

Компиляция программы

Рис. 3.4: Компиляция программы

3.3 Расширенный синтаксис командной строки NASM

Выполняю команду, указанную на (рис. 3.5), компилируется исходный файл hello.asm в obj.o, расшиерние .o говорит о том, что файл - объектный, помимо него флаги -g -l подготвоят файл отладки и листинга соответственно.

Возможности синтаксиса NASM

Рис. 3.5: Возможности синтаксиса NASM

3.4 Компоновщик LD

Далее передаю объектный файл компоновщику, с помощью команды ld. (рис. 3.6)

Отправка файла компоновщику

Рис. 3.6: Отправка файла компоновщику

Выполняю следующую команду, результатом будет созданный файл main, скомпонованный из объектного файла obl.o. (рис. 3.7)

Создание исполняемого файла

Рис. 3.7: Создание исполняемого файла

3.5 Запуск исполняемого файла

Запускаю исполняемый файл для проверки. (рис. 3.8)

Запуск программы

Рис. 3.8: Запуск программы

3.6 Задания для самостоятельной работы

Создаю копию файла для последующей работы с ней. (рис. 3.9)

Создание копии

Рис. 3.9: Создание копии

Редактирую копию файла, поменяв ‘Hello, world!’ на свое имя и фамилию Транслирую копию файла в объектный файл, компоную и запускаю. (рис. 3.10)

Проверка работоспособности скомпонованной программы

Рис. 3.10: Проверка работоспособности скомпонованной программы

Убедившись в корректности работы программы, копирую рабочие файлы в свой локальный репозиторий. (рис. 3.11)

Отправка файлов в локальный репозиторий

Рис. 3.11: Отправка файлов в локальный репозиторий

Загрузка изменений на свой репозиторий на GitHub. (рис. 3.12)

Загрузка изменений

Рис. 3.12: Загрузка изменений

4 Выводы

При выполнении данной лабораторной работы я освоила процедуры компиляции и сборки программ, написанных на ассемблере NASM.

5 Список литературы

2. Курс на ТУИС
3. Лабораторная работа №4